ESTGENOME_GPU

Nathan Weeks
Com S 590
Dec. 15, 2009
Task

Align (short) “EST” sequences (generated from cDNA, which is generated from mRNA) to (long) genomic sequences.

Possible reasons include:

- Find genes
- Measure gene expression levels (with “RNA-Seq data; many short sequences)
  - What genes are “on”, and to what extent?
1. Do linear-space, quadratic-time Smith-Waterman “scan” (no backtracking data stored) to find beginning & end positions of optimal alignment in both EST and genome sequence (>99% of runtime for “real” sequences)

2. Do quadratic-space, quadratic-time Needleman-Wunch (global alignment) on the resulting subsequences to get the actual alignment.
EST_GENOME_GPU

1. Do linear-space, quadratic-time Smith-Waterman “scan” (no backtracking data stored) to find beginning & end positions of optimal alignment in both EST and genome sequence (>99% of runtime for “real” sequences)

2. Do quadratic-space, quadratic-time Needleman-Wunch (global alignment) on the resulting subsequences to get the actual alignment.
Data Dependencies

Genome sequence

Source: "Biological Sequence Alignment on Graphics Processing Units" [Voss et al]
Algorithm Details

Given query sequence of length M and genome of length N, runtime of Smith-Waterman scan is O(MN), and space required is O(CM), where C is constant

- One query sequence per kernel launch (artificial limitation for prototype)
  - One thread block per query sequence (synchronization required after each “iteration” across genome sequence)
    - One thread per EST (query) sequence base (limits query length to 512, the maximum number of threads in a thread block)
    - Three shared-memory buffers length of query sequence (+1) to store temporary results
    - Six shared-memory buffers length of query sequence (+1) to store information regarding best alignment seen at that query position
<table>
<thead>
<tr>
<th>Genome</th>
<th>A</th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genome</td>
<td>A</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>G</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>G</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

**Genome Diagram**

- E: Green
- S: Yellow
- T: Red
- G: Blue
- C: Pink
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
- G: White
- C: Purple
- A: Yellow
- T: Green
- G: Red
- C: Blue
- A: Orange
- T: Black
<table>
<thead>
<tr>
<th>Genome</th>
<th>A</th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>C</th>
<th>A</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>G</td>
<td>*</td>
<td>X</td>
<td>*</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>E</td>
<td>A</td>
<td>F</td>
<td>7</td>
<td>4</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>S</td>
<td>C</td>
<td>*</td>
<td>T</td>
<td>3</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>T</td>
<td>A</td>
<td>3</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genome</td>
<td></td>
<td>A</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>G</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S C T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T A S C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genome</td>
<td>A</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>G</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>G</td>
</tr>
<tr>
<td>--------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>G</td>
<td>*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E A</td>
<td>*</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S C</td>
<td>*</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T A</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Genome</td>
<td>A</td>
<td>C</td>
<td>T</td>
<td>G</td>
<td>G</td>
<td>A</td>
<td>C</td>
<td>A</td>
<td>A</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td>*</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E A</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scoring Matrix - Problem

- Read-only lookup table set before kernel launch
  
  ```c
  int scoring_matrix[256][256]
  ```

- Indexed using DNA “characters”
  
  ```c
  scoring_matrix[est_base][genome_base]
  ```

**Problem:** 64 KB

- size of constant memory (64KB on G80)
- > size of constant cache (8KB per SM on G80)

*Accessed often; global memory too slow!*
Scoring Matrix - Solution

DNA sequence ASCII values:

- A = 0x41 = 0100 0001
- C = 0x43 = 0100 0011
- G = 0x47 = 0100 0111
- T = 0x54 = 0101 0100
- N = 0x4e = 0100 1110 (unknown base)

Notice that the lowest-order 3 bits are unique

- Use more compact scoring matrix:

  __constant__ char scoring_matrix[8][8]

- 64 bytes – easily fits in constant cache
  Index with lowest-order 3 bits of sequence:

  scoring_matrix[est_base&0x07][genome_base&0x07]
Highest-scoring alignment

To conserve shared memory, store some per-thread information in registers (variables) that will be reported for only the thread that found the highest-scoring alignment.

```c
short score;       // max alignment score for this thread
int score_thread; // max score & threadnum packed into 4b
__shared__ int max_score_thread;
max_score_thread = 0;
...
score_thread = (int)score << 16 | threadIdx.x;
atomicMax(max_score_thread, score_thread);
__syncthreads(); // wait for other threads
if (max_score_thread == score_thread)
// this thread found the highest-scoring alignment;
// its set of alignment results is the output of kernel
```
**SM Occupancy**

```bash
$ nvcc --ptxas-options -v -arch \ 
    sm_12 -c \ 
    kernel_smith_waterman_scan.cu
ptxas info: Compiling entry function 'smith_waterman_scan'
ptxas info: Used 30 registers,
12380+12376 bytes smem, 576 bytes cmem[0], 24 bytes cmem[1]
```

Due to shared memory (Note: hard-coded in prototype) & register usage, total length of all query sequences processed per-SM cannot exceed 512bp (though this could be several short-read, ”RNA-Seq” sequences).

<table>
<thead>
<tr>
<th>GPU Occupancy Data is displayed here and in the graphs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Threads per Multiprocessor</td>
</tr>
<tr>
<td>Active Warps per Multiprocessor</td>
</tr>
<tr>
<td>Active Thread Blocks per Multiprocessor</td>
</tr>
<tr>
<td>Occupancy of each Multiprocessor</td>
</tr>
</tbody>
</table>

**Physical Limits for GPU:**

<table>
<thead>
<tr>
<th></th>
<th>1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threads / Warp</td>
<td>32</td>
</tr>
<tr>
<td>Warps / Multiprocessor</td>
<td>32</td>
</tr>
<tr>
<td>Threads / Multiprocessor</td>
<td>1024</td>
</tr>
<tr>
<td>Thread Blocks / Multiprocessor</td>
<td>8</td>
</tr>
<tr>
<td>Total # of 32-bit registers / Multiprocessor</td>
<td>16384</td>
</tr>
<tr>
<td>Register allocation unit size</td>
<td>512</td>
</tr>
<tr>
<td>Shared Memory / Multiprocessor (bytes)</td>
<td>16384</td>
</tr>
<tr>
<td>Warp allocation granularity (for register allocation)</td>
<td>2</td>
</tr>
</tbody>
</table>

**Allocation Per Thread Block**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Warps</td>
<td>16</td>
</tr>
<tr>
<td>Registers</td>
<td>15360</td>
</tr>
<tr>
<td>Shared Memory</td>
<td>12800</td>
</tr>
</tbody>
</table>

These data are used in computing the occupancy data in blue

**Maximum Thread Blocks Per Multiprocessor**

<table>
<thead>
<tr>
<th>Limited by Max Warps / Multiprocessor</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited by Registers / Multiprocessor</td>
<td>2</td>
</tr>
<tr>
<td>Limited by Shared Memory / Multiprocessor</td>
<td>1</td>
</tr>
</tbody>
</table>

**Thread Block Limit Per Multiprocessor highlighted** RED

CUDA GPU Occupancy Calculator (source: NVIDIA)
Bonus - ATI Radeon HD 5750

Features
- 1 GB global memory
- 32KB “LDS” memory (equivalent to NVIDIA shared memory)
- 9 SIMD engines (equivalent to NVIDIA SM)
- 16 thread processors (equivalent to NVIDIA SP) per SIMD engine. Each thread processor has 5 ALUs.

Limitations (in beta driver)
- Inaccessible from OpenCL:
  - 64KB L2 texture caches
  - 8KB L1 texture caches
  - 64KB Global Data Share
- Limited access in OpenCL:
  - 16KB LDS memory/work group
  - 256MB global memory/work group
  - 256 work-items/work group (hardware supports 1024)
OpenCL Implementation

- CUDA kernel translated to OpenCL
- Kernel executed from PyOpenCL
  - Due to buggy ATI Catalyst drivers for Linux, Windows XP used; didn't have time to port POSIX-based EST GENOME code to Windows, so used PyOpenCL to invoke kernel
- Naïve implementation didn't attempt to exploit Radeon VLIW architecture (5 ALUs per thread processor)

To estimate timings:
- Only ran kernel implementing Smith-Waterman scan, not followup Needleman-Wunch alignment on the CPU to get actual alignment (note that this typically took less than a second on the CPU anyway)
- Only ran one Smith-Waterman scan, then took resulting time times 3 to mimic default three Smith-Waterman scans performed by EST GENOME

Bottom line: timings extrapolated – take with a grain of salt – but probably are in the ballpark of a “real” implementation
EST_GENOME_GPU Performance

Alignment against Medicago truncatula chromosome 6 (approx 23M bp)

CPU results generated with original EST_GENOME program compiled with gcc 4.1.2 using aggressive optimization ("-O3").

ATI timings extrapolated from running an OpenCL kernel once and multiplying the resulting time by 3. No ATI results for 512bp query sequence, as beta OpenCL implementation limits work-group size to 256 work-items. The hardware can support 1024 work-items per work-group.
Possible Improvements

- Reverse and both forward scans can be done simultaneously on GPU (three thread blocks); easy 3X speedup for a single query sequence
- Merge genome sequence and splice site arrays (genome sequence lower-order nibble of a byte, splice site flag upper-order nibble) to reduce global memory accesses to only a single array the length of the genome sequence
- Use shared-memory buffer for genome sequence with remaining shared memory – didn't improve performance when I tried it (bank conflicts? Buffer not big enough? Bugs in my code?)

**Biggest performance boost for “real-world” applications**: process multiple query sequences simultaneously.

- Naive implementation used only a single SM (out of 30) on the NVIDIA GTX 280; could align up to 512bp of query-sequence per SM; e.g.:
  - Two 250bp “454” reads per SM (60 simultaneously per GPU)
  - Four 100bp Illumina sequences per SM (120 per GPU)
  - Eight 50bp SOLiD reads per SM (240 per GPU)
Questions?